Tool Integration

Serge Autexier and Dieter Hutter

23rd April 2004

What's the Goal?

- We want to use tools in the courses
- What kind of course, for what to use the tool?
- How to implement the combination of course and tool?

Kinds of Course Documents

- Content Header of Sections, Subsections, Paragraphs
- Outline More abstract, shorter than content
- Lecture Slides
- Lecture Notes Slides + Annotations by students
- Course booklet-like, with clear relationship to lecture structure

From Document Structuring Facilities in MMISS, Krieg-Brückner et. al.

Use of Course Documents

	Paper **Text+Pictures**	Hyper **Hyper-Medium**	Replay **Replay in Tool**	Interactive Interaction
Contents	–	–	–	–
Outline	–	–	–	–
Lecture in class, annotated	handout before lecture	laptop browsing during lecture	demonstrate use of tool	experiment in tool
Lecture Notes after presentation, self-contained	handout after lecture	offline browsing personal annotations	demonstrate examples	experiment with examples
Course self-study	course script personalized?	personal navigation dynamic?	model solutions	personal solutions

From Document Structuring Facilities in MMISS, Krieg-Brückner et. al.

Roles of Tools

- **Object of teaching** (Your tool(s)!)

  Teaching CATS, Isabelle, IsaWin, VSE, MAYA, . . .

- **Visualization** (special purpose, convenience)

  Use emacs to display files, DaVinci for graphs, . . .

- **Service** (compiler, CAS, . . .)

  Check answers of students, translate between representations, compiling, . . .

Requirements to Tools by Usage-Type

- **Object of teaching**
  - Interactive Demonstration
    - Need possibility to initialize tool as wanted
  - Automatic Demonstration
    - Need possibility to run scripts with pauses/stepping
  - For Exercises
    - Need possibility to initialize the tool
    - Need mechanism to check when exercise is finished
    - Supervised Exercises: return feedback
- **Visualization**
- **Providing support**

Techniques for Tool Integration

- Direct calls via shell-scripts
  - Simple if you know scripting language
  - heavily context dependent, bad for reuse
  - Need to know both the tool and where it is installed
  - Simple to combine with PDF Slides and Acrobat Reader
  - Probably good for standalone distributions (pack slides, shell-scripts, and tools).

Techniques for Tool Integration (cont'd)

- By brokering mechanisms
  - Similar to shell-scripts, but no local installation necessary
    - Separation between tool installation and its use
    - As author of the lecture, you only need to know the tool
    - Less context dependent, better for reuse
    - Requires a broker (someone to set it up)
    - Less suitable for standalone distributions
    - More suitable for distributions with remote access
      (e.g. centrally installed tools for the whole univ. department)
  - More abstract descriptions under development

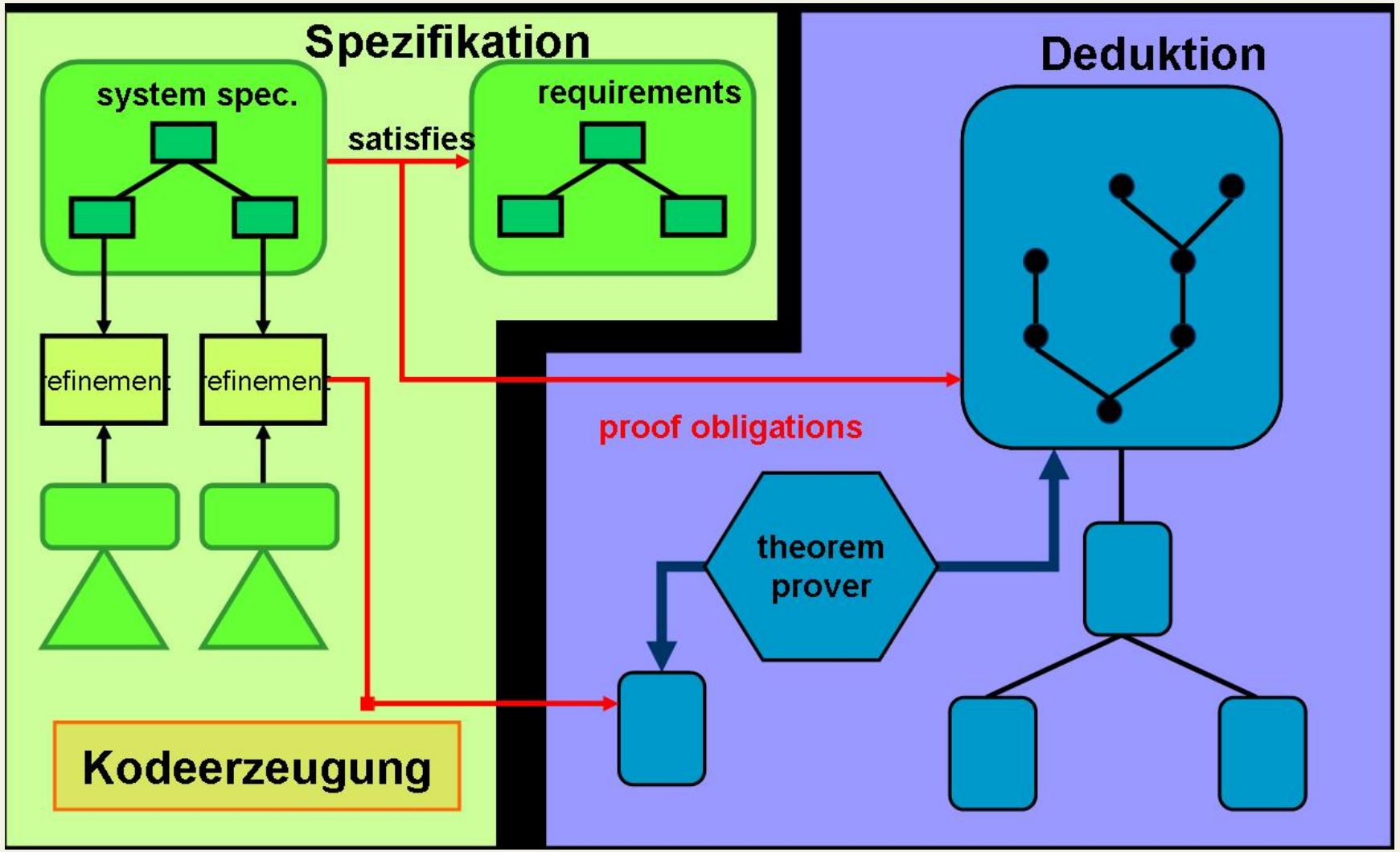
(Some) Integrated Tools

- Tools for Visualisation
  - Emacs
  - Hets
- Tools for Services
  - Casl Tool Set (CATS)
  - Compiler for programming languages (e.g. Haskell)
- Tools as objects of teaching
  - VSE
  - MAYA

Verification Support Environment (VSE)

- Developed on behalf of the German Bundesamt für Sicherheit in der Informationstechnik (1991–1999)
- Tool to support formal software development
- Support for specification and verification of sequential and concurrent programs
- Support for all phases of the development (requirement analysis – automated code generation)

VSE Inside


Spezifikation
Deduktion
system spec.
requirements
satisfies
refinement
refinement
proof obligations
theorem prover
Kodeerzeugung

A Sample Specification in VSE-SL

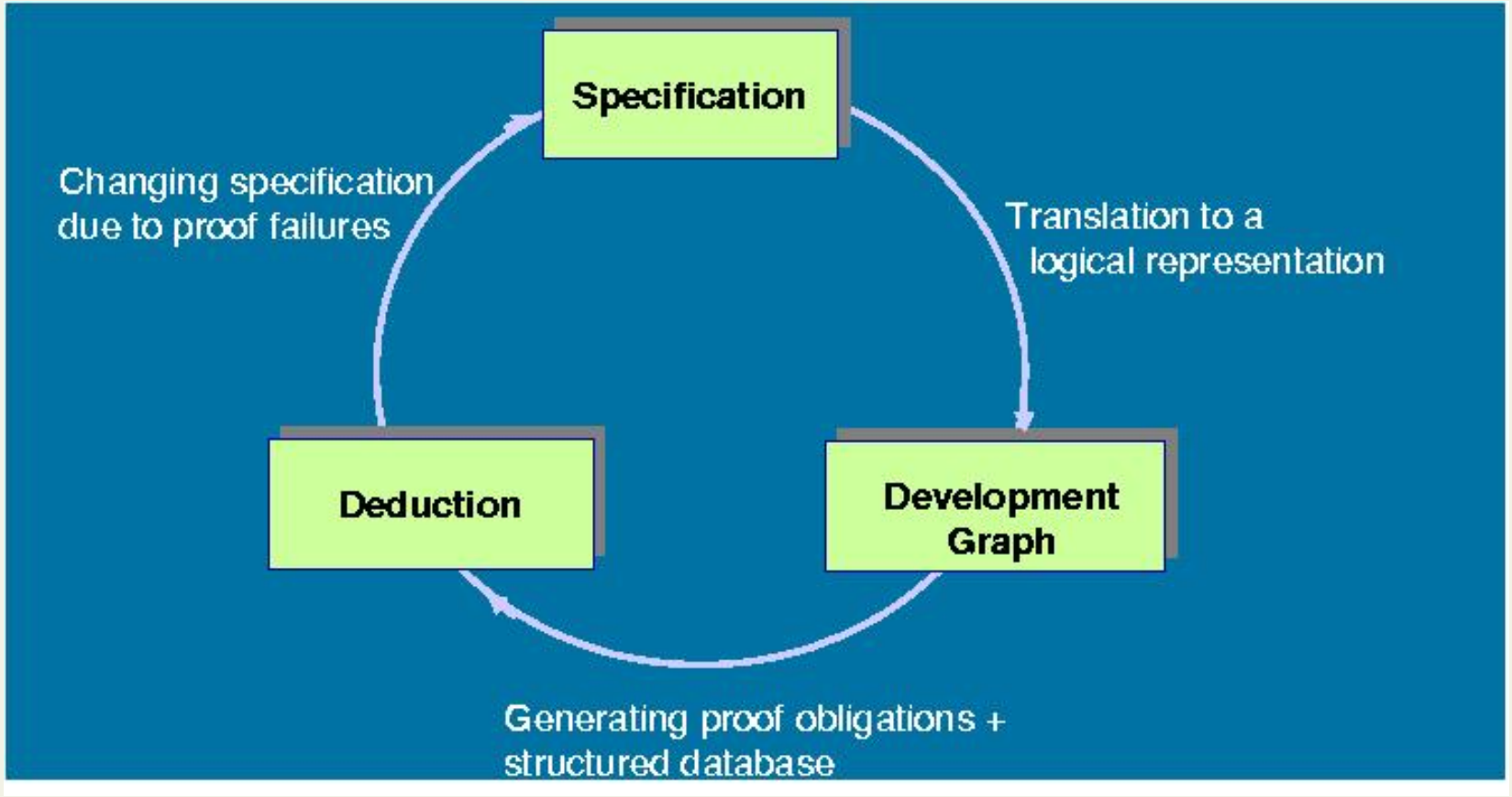
```
THEORY natlist
 USING NATURAL
 TYPES list = FREELY GENERATED BY empty | cons(head : nat,tail : list)

 FUNCTIONS length : list -> nat;
           append : list, list -> list
 VARS
  l, l1, l2: list;
  n: nat

 AXIOMS
  FOR length:
   DEFFUNC
    length(l) =
     SWITCH l IN
      CASE empty: 0
      CASE cons: succ(length(tail(l)))
     NI
```

```
  FOR append:
   DEFFUNC
    append(l1,l2) =
     SWITCH l1 IN
      CASE empty: l2
      CASE cons:  cons(head(l1),append(tail(l1),l2))
     NI
 SATISFIES natlistproperty
THEORYEND

[...]
```

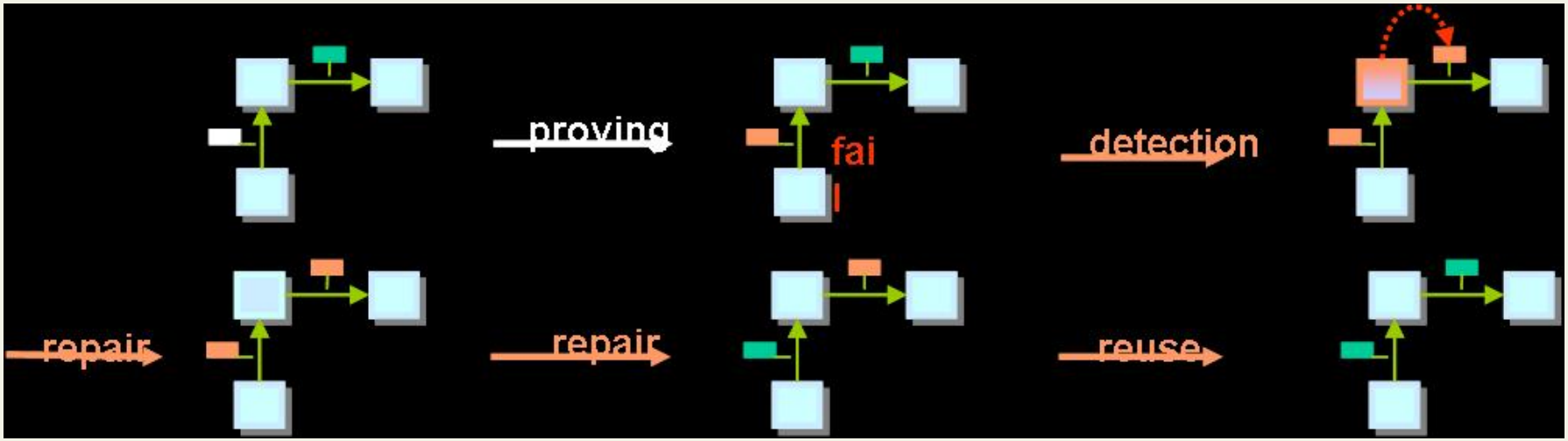
To use VSE on a given sample specification to do some demonstration or exercise (no feedback) please click here.

MAYA - Management of Change

Grews out of the need for even more sophisticated management of change

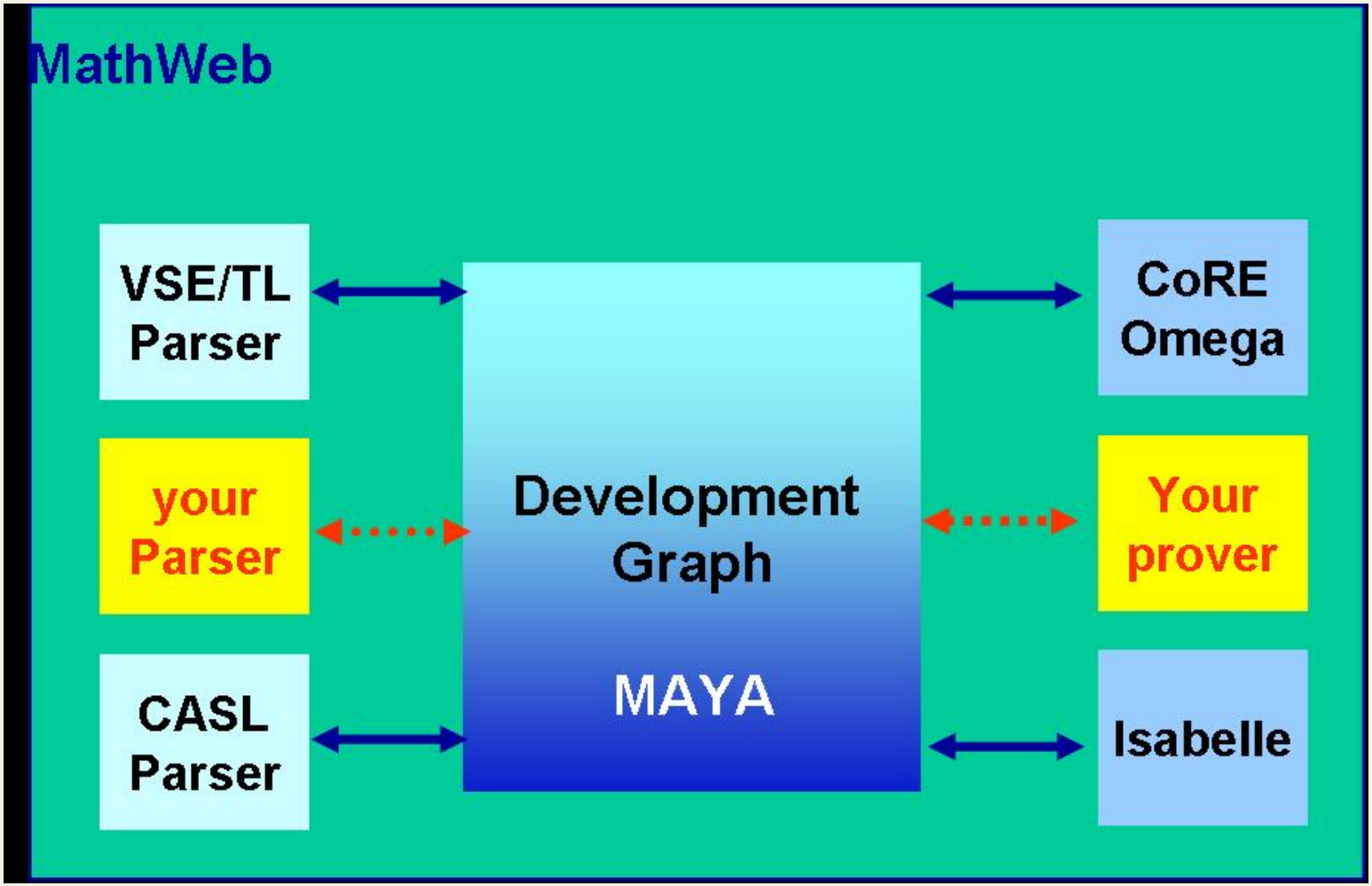


MAYA - Management of Change



- Uniform formalism for structured formal specifications
- Sophisticated management of change by
  - Decomposition of formal relationships to reduce amount of proof obligations
  - Difference analysis between two versions to reuse existing decompositions and proofs

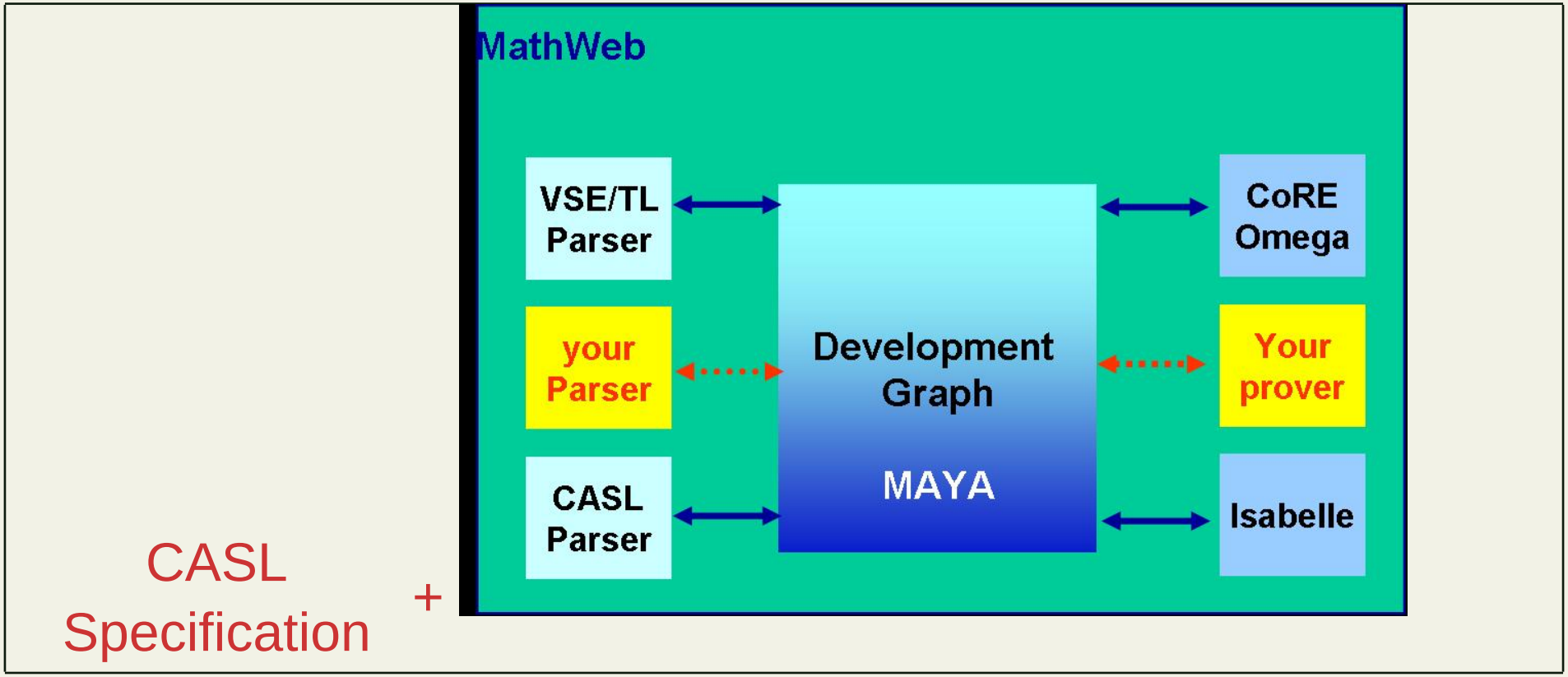
MAYA's Social Life



A CASL Specification

```
spec list =
  { sort elem; }
then
  {
   generated type list ::= nil | cons(fst:elem; rest:list);

   var l1,l2:list;
   var n1,n2:elem;

   op app  : list * list -> list, assoc, unit nil;

   axiom app(cons(n1,l1),l2) = cons(n1, app(l1,l2));

   op addlast(n:elem; l:list):list =
      cons(n,nil) when l = nil
      else cons(fst(l), addlast(n,rest(l)));
  }

[...]
```

Teaching MAYA with CASL...



Conclusion

- Combination of course and tool can be very complex

  (Depending on kind of course and role of tools)

- What we have
  - Good and easy-to-use approach for simple tool calls via shell-scripts

    (if you like latex and unix/linux)

    Integrated different tools without to much initial effort
  - Good for lectures/lecture notes with tool demos/unsupervised exercises
  - Good to persuade authors to use MMiSSLatex

A Wish List

- Increase reusability of slides, scripts, and tools without loosing easy-to-write/use feature?
  - Standard(s) (language + semantics) to describe tool integration scenarios
- How to have an easy-to-write and use for more sophisticated tool interactions? Supervised learning?
  - Standard(s) (language + semantics) to describe exercises and supervision/tutoring for students